KOKKA & BACKUS, PC

Facsimile

DATE October 7, 2009

	NAME	FAX	PHONE
TO	Ben C. Wang Examiner U.S. Patent and Trademark Office	(571) 270-2240	(571) 270-1240
FROM	Charisse E. Leong Patent Paralegal Kokka & Backus, PC	(650) 566-9922	(650) 566-9912

PAGES (INCLUDING THIS COVER PAGE): 09

RE U.S. Patent Application No. 10/711,148

Message:

Please find attached the Proposed/Unofficial/Not for Entry Claim Listing for U.S. Patent Application No. 10/711,148.

Sincerely,

[illegible]

Charisse E. Leong
Patent Paralegal

CLAIM LISTING

1. (Currently amended) In a form-based development system, a method for dynamically constructing a form under an object framework during development of an application by a user, the method comprising:

providing an ancestor class under [[an]]the object framework, the ancestor class for representing a form in the development system ~~and configured to allow migration of one or more tools and one or more programs from another form-based development system to the development system~~;

in response to user input, creating a descendant class inheriting from the ancestor class for representing a particular form to be included in the application without directly manipulating metadata of the ancestor class[[form]];

generating intermediate language instructions for creating methods of the descendant class under the object framework;

creating a type delegator for the descendant class, the type delegator being configured to persist information associated with the particular form and to intercept the metadata using a metadata application programming interface call, the metadata being manipulated to change a class type associated with the type delegator dynamically and to create a data structure to track changes to be instantiated with the particular form and the descendant class being configured to act as a proxy configured to represent the particular form, thereby enabling the descendant class to track changes made to the particular form during development of the application;

creating an instance of the descendant class;

constructing the particular form in the development system based on the instance of the descendant class and making a design time representation of the particular form visible to the user without using source code and without compiling the descendant class;

tracking changes to the particular form made during development of the application using the type delegator, the type delegator being configured to persist the information to the descendent class without changing the particular form during development of the application prior to runtime; and

PROPOSED/UNOFFICIAL/NOT FOR ENTRY

~~persisting information regarding the particular form; and~~

subsequently, generating a version of the particular form at runtime based on [[the]]persisted information.

2. (Previously presented) The method of claim 1, wherein the object framework comprises an infrastructure for building, deploying and running applications having a common language runtime.

3. (Currently amended) The method of claim 1, wherein said creating step includes creating [[a]]the descendant class [[for]]configured to represent[[ing]] the particular form in a user interface of the development system.

4. (Original) The method of claim 1, wherein said creating step includes inheriting a set of components provided by the ancestor class for representing components that may placed on the particular form.

5. (Original) The method of claim 1, wherein said creating step includes creating an assembly for the descendant class.

6. (Currently amended) The method of claim 1, further comprising:

creating another ~~second~~ descendant class which inherits from the descendant class, the created another~~second~~ descendant class for representing a form which inherits from the particular form.

7. (Original) The method of claim 1, wherein said constructing step includes constructing the particular form based upon the descendant class in a user interface of the development system.

8. (Original) The method of claim 1, wherein said constructing step includes displaying a component palette including components which the user can select for placement on the particular form.

PROPOSED/UNOFFICIAL/NOT FOR ENTRY

9. (Original) The method of claim 8, further comprising:
receiving user input for placing components selected from the palette on the particular form.

10. (Original) The method of claim 9, wherein the type delegator tracks creation of components on the particular form in response to a user placing a component on the particular form.

11. (Currently amended) The method of claim 9, wherein the type delegator persists information regarding components placed on the particular form, ~~thereby~~ enabling the components placed on the particular form to be recreated at runtime.

12. (Original) The method of claim 1, wherein said generating step includes generating a constructor for the descendant class.

13. (Original) The method of claim 12, wherein said step of generating a constructor includes generating intermediate language instructions for building the constructor.

14. (Original) The method of claim 13, wherein said step of generating intermediate language instructions includes using classes provided by the object framework for generating intermediate language instructions.

15. (Original) The method of claim 13, wherein said generating step includes generating instructions for calling the constructor of the ancestor class, thereby ensuring execution of an appropriate constructor implemented by the ancestor class.

16. (Original) The method of claim 1, wherein said generating step includes generating methods for overriding notification methods of the ancestor class.

17. (Original) The method of claim 16, wherein said generating step includes generating intermediate language instructions for overriding notification methods of the ancestor class.

18. (Original) The method of claim 1, wherein the type delegator provides information for enumerating fields, methods, properties, and events in response to user input on the particular form in the development system.

19. (Currently amended) The method of claim 1, wherein the type delegator generates the metadata ~~information~~ in response to user input on the particular form.

20. (Currently amended) The method of claim 19, wherein said step of generating the metadata ~~information~~ includes adding a reference to methods of the application assigned to components on the particular form.

21. (Currently amended) The method of claim 1, further comprising:
persisting a state of the particular form, enabling the particular form to be recreated at runtime.

22. (Original) The method of claim 21, wherein said persisting step includes persisting user input on the particular form.

23. (Original) A computer-readable medium having processor-executable instructions for performing the method of claim 1.

24. (Original) A downloadable set of processor-executable instructions for performing the method of claim 1.

25. (Currently amended) A development system for dynamically constructing a form responsive to user input under an object framework during development of an application, the system comprising:

PROPOSED/UNOFFICIAL/NOT FOR ENTRY

a computer having a processor and memory;

an ancestor class for representing the form under the object framework ~~and configured to allow migration of one or more tools and one or more programs from another form based development system to the development system~~;

a proxy module for creating a descendant class inheriting from the ancestor class in response to user input, dynamically generating methods of the descendant class, and constructing an instance as a particular form of the descendant class under the object framework for representing the form in the development system;

a type delegator for the descendant class for tracking user input on the particular form during development of the application, the type delegator being configured to persist information associated with the particular form and to intercept metadata using a metadata application programming interface call, the metadata being manipulated to change a class type associated with the type delegator dynamically and to create a data structure to track one or more changes to be instantiated with the particular form and the descendant class being configured to act as a proxy configured to represent the particular form, enabling the descendant class to track the one or more changes made to the particular form during development of the application;

a persistence mechanism for persisting user input on the particular form and configured to persist the information from the type delegator to the particular form without changing the particular form during development of the application prior to runtime; and

a module for displaying a design time representation of the particular form in a user interface of the development system based on the descendant class and the persisted user input without using source code and without compiling the descendant class and subsequently generating a version of the particular form at the runtime based on the persisted information.

26. (Previously presented) The system of claim 25, wherein the object framework comprises an infrastructure for building, deploying and running applications having a common language runtime.

27. (Original) The system of claim 25, wherein the proxy module creates an assembly for the descendant class.

28. (Currently amended) The system of claim 25, wherein the proxy module creates a descendant class for representing the particular form in a user interface of the development system.

29. (Currently amended) The system of claim 28, wherein the descendant class inherits a set of components provided by the ancestor class for representing components that may placed on the particular form.

30. (Original) The system of claim 28, wherein the proxy module creates a second descendant class which inherits from the descendant class, the created second descendant class for representing a second form which inherits from the particular form previously created in the development system.

31. (Currently amended) The system of claim 28, wherein the particular form includes a component palette comprising components which the user can select.

32. (Currently amended) The system of claim 31, wherein the type delegator persists user input for placing components selected from the palette on the particular form.

33. (Currently amended) The system of claim 32, wherein the type delegator persists information regarding components placed on the particular form, ~~thereby~~ enabling components to be recreated at runtime.

34. (Original) The system of claim 25, wherein the proxy module generates a constructor for the descendant class.

35. (Original) The system of claim 34, wherein the proxy module generates intermediate language instructions for building the constructor.

36. (Original) The system of claim 35, wherein the proxy module generates intermediate language instructions using classes for generating intermediate language instructions provided by the object framework.

37. (Original) The system of claim 35, wherein the proxy module generates instructions for calling the constructor of the ancestor class, thereby ensuring execution of an appropriate constructor implemented by the ancestor class.

38. (Original) The system of claim 25, wherein the proxy module generates methods for overriding notification methods of the ancestor class.

39. (Original) The system of claim 38, wherein the proxy module generates intermediate language instructions for overriding notification methods of the ancestor class.

40. (Original) The system of claim 39, wherein the proxy module generates intermediate language instructions using classes for generating intermediate language instructions provided by the object framework.

41. (Currently amended) The system of claim 25, wherein the ~~type delegator provides~~ information [[for]]is used to enumerate[[ing]]e fields, systems, properties, and events in response to user input on the particular form during development of the application.

42. (Currently amended) The system of claim 25, wherein the type delegator generates the metadata ~~information~~ in response to user input on the particular form.

43. (Currently amended) The system of claim 42, wherein said metadata ~~information~~ includes a reference to methods of the application assigned to particular components on the particular form.

44. (Currently amended) The system of claim 43, wherein said metadata ~~information~~ and the

descendant class are used to reconstruct the particular form as part of the application at runtime.

45. (Currently amended) The system of claim 25, wherein the particular form comprises another form open on a visual design surface of the development system.

46. (Currently amended) The system of claim 25, wherein the persisting mechanism persists state of the particular form.

47. (Canceled)

48. (Currently amended) The system of claim 46, wherein the persisting mechanism enables the particular form to be recreated at runtime as part of the application.